Package: tracee (via r-universe)

November 24, 2026

Type Package

Title Easily Save Output and Trace it Back to Code

Version 0.1.0

Maintainer Philip Delff <philip@delff.dk>

Description Write output (plots and tables) ensuring traceability back to code. Includes a graphics saver with simple automation of stamping with source, destination and creation time. A list of plots can be saved at once. A user-friendly selection of output dimensions for presentations, on-screen inspections, and more available.

License MIT + file LICENSE

URL https://nmautoverse.github.io/tracee/, https://github.com/NMautoverse/tracee

BugReports https://github.com/NMautoverse/tracee/issues

Roxygen list(markdown = TRUE,roclets=c(``collate'', ``namespace'', ``rd'', ``vignette''))

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports NMdata (>= 0.2.4), data.table, ggplot2, grid, gridExtra, flextable, patchwork, ragg

VignetteBuilder knitr

Suggests testthat, withr, knitr, rmarkdown, magick

Encoding UTF-8

Config/testthat/edition 3

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev

Repository https://nmautoverse.r-universe.dev

Date/Publication 2026-11-24 13:44:48 UTC

RemoteUrl https://github.com/nmautoverse/tracee

RemoteRef HEAD

RemoteSha ba33378c3bf7dd20082d578273284cc68a1d3f80

Contents

canvasSize . . . 2
datwrite . . . 3
ftstamp . . . 4
ftwrite . . . 5
ggstamp . . . 8
ggwater . . . 10
ggwrite . . . 11
is.gg . . . 15
is.listnotplot . . . 16
lwrite . . . 16
padZeros . . . 20
pathFormat . . . 21
stampFlextab . . . 24
traceit . . . 24
writer . . . 26

Index **30**

`canvasSize` *Get predefined canvas sizes*

Description

Get predefined canvas sizes

Usage

```
canvasSize(canvas, scale = 1, simplify = TRUE)
```

Arguments

`canvas`	Can either be a character with one of the values • "standard" A powerpoint standard size, and two can be shown side by side - 7.2 by 5.4 in (WxH). • "wide" For a single wide plot on a powerpoint slide - 9.6 by 5.4 in • "A4" A full A4 page - 5.4 by 7.2 in • "square" As reads - 5.4 by 5.4 in • "wide-screen" For full screen display - 18.6 by 9 in or it can be a list with elements width and height with single values (unit is inches). Example: canvas=list(height=5,width=9).
`scale`	A scale to apply to both directions of the canvas size. This can be useful in combination with the pre-defined canvas sizes.
`simplify`	If only one canvas returned and simplify=TRUE, the result will not be wrapped in a list. For programming, you most likely want simplify=FALSE to be sure to always get the same format back.

Value

A list with numerical elements width and height as used by ggwrite.

See Also

Other Plotting: `ggstamp()`, `ggwater()`, `ggwrite()`

`datwrite` *Write data to file(s) using NMwriteData*

Description

This function writes data objects to files, with support for lists of data objects. It wraps NMwriteData and provides additional functionality for file path construction and metadata stamping.

Usage

```
datwrite(
  x,
  file,
  script,
  time,
  model,
  formats,
  path.format = NULL,
  subdir = NULL,
  quiet = FALSE,
  fun.path = NULL,
  ...
)
```

Arguments

`x`	A data frame or list of data frames to write
`file`	Character string specifying the output file name
`script`	Optional character string specifying the script name for stamping
`time`	Optional time stamp to include in output
`model`	Optional model object or path for organizing output files
`formats`	Character vector of output formats (passed to NMwriteData)
`path.format`	Character string or function defining the output path. Passed to `pathFormat`. Default (`NULL`) uses `file` as is.
`subdir`	Optional subdirectory for organizing output files
`quiet`	Logical. If `TRUE`, suppresses messages about ignored `model`/`script` arguments from `path.format`.
`fun.path`	Deprecated. Use `path.format`.
`...`	Additional arguments passed to NMwriteData

Value

A list of results from NMwriteData calls

`ftstamp` *Stamp and write flextab objects to one or multiple formats*

Description

Stamp and write flextab objects to one or multiple formats

Usage

```
ftstamp(
  ft,
  file,
  script,
  time,
  model,
  stamp.ft.style,
  stamp.format = NULL,
  stamp.funs = NULL
)
```

Arguments

`ft`	a flextable object
`file`	The file that the flextable will be written to (no file is written by this function)
`script`	path to script - will be pasted as caption.
`time`	The default is to insert a time stamp taken from result of Sys.time(). Using the time argument you can overrule this by setting a fixed string instead. Use "" to omit.
`model`	Character string or model object specifying the model name/identifier. Used for stamping. If a list (model object) is provided, the `$label` element is extracted.
`stamp.ft.style`	A list specifying formatting options for the stamp. Valid elements include: `size` (font size, default 4), `align` (text alignment, default "right"), `space` (line spacing, default 1), `bg` (background color, default "#ffffff"), and `color` (text color, default "#666666"). Any unspecified elements use the defaults.
`stamp.format`	A template for the stamp text using the placeholders `%script`, `%time`, `%model`, and `%file`, with `"\n"` for line breaks. Default is `"%file\n%script\n%model\n%time"`. See `createStamp`.
`stamp.funs`	Optional named list of functions (elements `script`, `time`, `model`, `file`) used to preprocess the values before they are inserted into `stamp.format`. See `createStamp`.

Value

A flextable object with the stamp added as a footer line.

ftwrite *Write flextable objects to one or multiple file formats*

Description

Writes `flextable` objects to various file formats with optional stamping (caption with script name, timestamp, and model information). Can write single flextables or lists of flextables, with automatic file naming and directory structure organization.

Usage

```
ftwrite(
  ft,
  file,
  formats,
  save = TRUE,
  show = !save,
  quiet = FALSE,
  script,
  time,
  model,
  stamp.ft.style,
  path.format = NULL,
  subdir = NULL,
  use.names = TRUE,
  stamp.format = NULL,
  stamp.funs = NULL,
  fun.path = NULL
)

writeFlextab(ft, file, script, formats, save, quiet = FALSE, ...)
```

Arguments

`ft`	A `flextable` object or a named list of `flextable` objects. If a list is provided, each element will be written to a separate file with names derived from the list element names.
`file`	Character string specifying the base file name to save to (without extension). The actual extension(s) will be determined by the `formats` argument. If `ft` is a list and elements are named, those names will be appended to this base file name.
`formats`	Character vector specifying one or more output formats. Supported formats are: "png", "docx", "pptx", "html", "rds". Default is `NULL`, which will infer the format from the `file` extension.
`save`	Logical. Should the table be saved to file? Default is `TRUE`. Set to `FALSE` to only return/show the flextable without saving. Hint: if you use a flag variable (like `exportFlag`), you can use `save = exportFlag` to control output generation.

show	Logical. Should the flextable be printed/displayed? Default is the opposite of save (i.e., !save). Combining save = TRUE and show = TRUE in knitr documents can be useful to display the table in the rendered document while also saving high-quality files.
quiet	Logical. If TRUE, suppresses messages about files being written. Default is FALSE.
script	Character string or path to the script file. If provided, will be included in the table stamp/caption. This helps document which script generated the output.
time	Character string or POSIXct timestamp. The default behavior is to include a timestamp if script is provided. You can pass any string as time for a custom format, or use time = "" to omit the timestamp entirely.
model	Character string or model object specifying the model name/identifier. Used for stamping and for organizing output directories via path.format.
stamp.ft.style	A list of formatting options for the stamp, passed to ftstamp. Valid elements: size, align, space, bg, color.
path.format	Character string or function defining the output path. A template using %model, %script, and %name placeholders, or a function as returned by pathFormat. Passed to pathFormat. Default (NULL) uses file as is.
subdir	Character string specifying a subdirectory within the path structure. Used in conjunction with path.format.
use.names	Logical. If length(ft) > 1, should names(ft) be used in the output file names? Default is TRUE. If FALSE, numeric indices (1, 2, 3, ...) are used instead. Only used if save = TRUE and ft is a list.
stamp.format	A template for the stamp text using the placeholders %script, %time, %model, and %file, with "\n" for line breaks. Default is "%file\n%script\n%model\n%time". See createStamp.
stamp.funs	Optional named list of functions (elements script, time, model, file) used to preprocess the values before they are inserted into stamp.format. See createStamp.
fun.path	Deprecated. Use path.format.
...	Additional arguments passed to ftstamp for stamp customization.

Details

The function handles both single flextable objects and lists of flextables:

- For single flextables: writes to file(s) with name specified by file
- For lists of flextables: writes each element to a separate file, with names constructed from file and the list element names

When ft is a list:

- Named list elements use their names in the output file names
- Unnamed elements are automatically numbered with zero-padded indices

Multiple formats can be written simultaneously by specifying a vector of formats. Each format will be saved as a separate file with the appropriate extension.

Value

Invisibly returns the result from ftwriteOne (for lists, returns results for all written tables). If show = TRUE, also prints the flextable(s) to the current device.

Functions

- writeFlextab(): Deprecated function name. Use ftwrite.

File Formats

The following formats are supported via the flextable package save functions:

- "png": Image format via save_as_image
- "docx": Microsoft Word via save_as_docx
- "pptx": Microsoft PowerPoint via save_as_pptx
- "html": HTML format via save_as_html
- "rds": R data format (saves the flextable object itself)

See Also

ftwriteOne for writing a single flextable, ftstamp for adding stamps to flextables, writer for a unified interface to write various object types, pathFormat for organizing output directories, save_as_image, save_as_docx, save_as_pptx, save_as_html

Examples

```
## Not run:
library(flextable)

# Create a simple flextable
ft1 <- flextable(head(mtcars))

# Write to PNG (default format)
ftwrite(ft1, file = "mytable")

# Write to multiple formats
ftwrite(ft1, file = "mytable", formats = c("png", "docx", "html"))

# Write with stamping
ftwrite(ft1, file = "mytable",
        script = "analysis.R",
        model = "run001")

# Write a list of tables
ft_list <- list(
  summary = flextable(summary(mtcars)),
  head = flextable(head(mtcars)),
  tail = flextable(tail(mtcars))
)
ftwrite(ft_list, file = "tables", formats = "docx")
```

```
# Creates: tables_summary.docx, tables_head.docx, tables_tail.docx

# Use with path structure for model organization
ftwrite(ft1, file = "summary",
        model = "run001",
        path.format = "%model/%name_%model")
# Writes to: run001/summary_run001.png

## End(Not run)
```

---

ggstamp *Always stamp your plots with script name*

Description

This function is used to stamp ggplot type plots with datetime and script name. User must provide the script name.

Usage

```
ggstamp(
  plot,
  file,
  size,
  script,
  time,
  model,
  stamp.gg.style,
  stamp.format = NULL,
  stamp.funs = NULL
)
```

Arguments

`plot`	The plot to be stamped.
`file`	An optional output filename to be included in the stamp.
`size`	The size (list) for the plot, not the caption. Used to calculate the size of the caption.
`script`	the script name. Date and time will be added automatically.
`time`	The timestamp to be included.
`model`	An optional model name or object to be included in the stamp.
`stamp.gg.style`	A list of formatting options for the stamp text. For ggplot objects, valid elements include size, colour, hjust, vjust, and other parameters accepted by `ggplot2::element_text()`. For gtable objects, valid elements include col, fill, alpha, lty, lwd, lex, lineend, linejoin, linemitre, fontsize, cex, fontfamily, fontface, lineheight, and font. Default formatting is applied if not specified. Additionally, the element `position`

controls where the stamp is placed: "left", "center", "right" (default) or a number between 0 (left) and 1 (right). hjust controls justification of the lines within the stamp and defaults to 0 (left-justified). Example: stamp.gg.style=list(position="right", places the stamp on the right with left-justified lines.

stamp.format — A template for the stamp text using the placeholders %script, %time, %model, and %file, with "\n" for line breaks. Default is "%file\n%script\n%model\n%time". See createStamp.

stamp.funs — Optional named list of functions (elements script, time, model, file) used to preprocess the values before they are inserted into stamp.format. See createStamp.

Details

The stamp text is generated by createStamp based on stamp.format and stamp.funs. If the plot already has a caption, the stamp is added below it.

How the stamp is added depends on size:

- If size is given (always the case when called from ggwrite), the stamp is drawn in a separate panel underneath the plot (combined using patchwork). The panel height is calculated from the number of stamp lines and the font size, and the plot height is reduced accordingly so the total size matches size. Placement of the stamp (stamp.gg.style$position) and justification of its lines (stamp.gg.style$hjust) are controlled independently.
- If size is not given, the stamp is added as the plot caption (labs(caption=...)) and styled using theme(plot.caption=...). In this case, hjust controls both placement and justification.

For gtable objects (e.g. from gridExtra::arrangeGrob), the stamp is added at the bottom using arrangeGrob.

ggplot 2.2.1 or newer is required.

Value

A stamped plot object. If size is given, a patchwork object with the stamp panel underneath the plot. Otherwise the plot with the stamp added as caption. For gtable input, a gtable. If plot is a list, a list of stamped plots.

See Also

Other Plotting: canvasSize(), ggwater(), ggwrite()

Examples

```
library(ggplot2)
data(ChickWeight)
p1 <- ggplot(ChickWeight,aes(Time,weight,group=Chick,colour=factor(Diet)))+geom_line()
script <- "note"
ggstamp(p1,script)
## Or use ggwrite which will call ggstamp when the `script` argument is provided.
ggwrite(p1,script=script,canvas="wide")
```

ggwater *Add watermarks to ggplots*

Description

Add watermarks to ggplots

Usage

```
ggwater(text = "Not validated", scale = 1, rot = 30, col = "grey", alpha = 0.5)
```

Arguments

text	What should be written on the plot
scale	the size
rot	rotation of the mark. Don't remember the unit.
col	the colour (a value, not an expression) of the watermark. Default is grey.
alpha	alpha value for the watermark. Default is 0.5.

Details

This used to be based on https://www.r-bloggers.com/adding-watermarks-to-plots/ That solution stopped working, and this new solution is simpler and based on ggplot2 alone.

Value

a layer with a watermark that can be added to a plot

See Also

Other Plotting: canvasSize(), ggstamp(), ggwrite()

Examples

```
library(ggplot2)
ff <- qplot(1:10, 11:20) + ggwater()
```

ggwrite *Export ggplot plots (and related objects) to files with stamping support*

Description

Writes ggplot plots, gtable objects, or lists thereof to PNG or PDF files with optional stamping (caption with script name, timestamp, and model information). Supports flexible canvas sizing, multiple output formats, and both single-file and multi-file output for lists of plots.

Usage

```
ggwrite(
  plot,
  file,
  canvas = NULL,
  formats,
  onefile = FALSE,
  res = 200,
  paper = "special",
  save = TRUE,
  show = !save,
  use.names = FALSE,
  subdir = NULL,
  script,
  time,
  model,
  name.all.pdf,
  quiet = FALSE,
  path.format = NULL,
  method.png = c("ragg", "png"),
  stamp.format = NULL,
  stamp.funs = NULL,
  stamp.gg.style = NULL,
  useNames,
  fun.path = NULL
)
```

Arguments

plot	A plot object or a list of plot objects. Normally generated with ggplot or qplot. Can also be gtable objects from grid.arrange or arrangeGrob. Support for gtable is experimental.
file	Character string specifying the output file path. Must end in .png or .pdf. If missing and save = TRUE, an error is raised. If missing and save = FALSE, the plot is shown on screen instead. For lists of plots, see onefile for naming behavior.

canvas — Either a list with height and width elements (in inches), or a character string shortname of a predefined canvas size. Default is "standard". See canvasSize for available sizes and custom size specification. Multiple canvas sizes can be specified as a named list to generate outputs at different sizes.

formats — Character vector specifying file format(s) to write. Must be a subset of c("png", "pdf"). Default is to use the format matching the file extension. Multiple formats can be specified to write the same plot in different formats.

onefile — Logical. Only applicable if plot is a list. If TRUE, all plots are written to a single multi-page PDF (file must end in .pdf). If FALSE, numbered files are created - one per list element. Default is FALSE.

res — Numeric. Resolution in pixels per inch for PNG output. Default is 200. Passed to png.

paper — Character string specifying paper type. Only used with PDF device. Default is "special". See pdf for options.

save — Logical. Should the plot be saved to file? Default is TRUE. Set to FALSE to only display the plot without saving. Useful when controlled by a flag variable (e.g., save = writeOutputs).

show — Logical. Should the plot be displayed on screen? Default is the opposite of save (i.e., !save). Setting both save = TRUE and show = TRUE in knitr documents can display the plot in the rendered document while also saving high-quality files.

use.names — Logical. If length(plot) > 1, should names(plot) be used in the output file names? Default is FALSE, which uses numeric indices (1, 2, 3, ...). Only used if save = TRUE and plot is a list.

subdir — Character string specifying a subdirectory within the path structure. This is passed to the path function generated from path.format. Default is NULL.

script — Character string or path to the script file. If provided, a stamp (caption) will be added to the plot showing the script name, timestamp, and optionally the model. Requires ggplot2 >= 2.2.1.

time — Character string or POSIXct timestamp. By default, a timestamp is included if script is provided. You can pass a custom time string or use time = "" to omit the timestamp.

model — Character string or model object specifying the model name/identifier. Used for stamping and for organizing output directories via fun.path.

name.all.pdf — Character string specifying the filename to use when writing all plots to a single PDF file (when onefile = TRUE). Only used when file = "". Default is "all.pdf".

quiet — Logical. If TRUE, suppresses messages about files being written. Default is FALSE.

path.format — Character string or function defining the output path. A template using %model, %script, and %name placeholders, or a function as returned by pathFormat. Passed to pathFormat. Default (NULL) uses file as is.

method.png — Character string, one of "png" or "ragg", controlling which graphics device is used for PNG output. "png" uses the base png device; "ragg" uses agg_png, which produces output that renders more consistently across platforms (recommended for reproducible/snapshot-tested figures). Default is "png".

stamp.format	A template for the stamp text using the placeholders %script, %time, %model, and %file, with "\n" for line breaks. Default is "%file\n%script\n%model\n%time". See createStamp.
stamp.funs	Optional named list of functions (elements script, time, model, file) used to preprocess the values before they are inserted into stamp.format. See createStamp.
stamp.gg.style	A list of formatting options for the stamp text, passed to ggstamp. For ggplot objects, elements accepted by ggplot2::element_text() (e.g. size, colour, hjust).
useNames	Deprecated. Use use.names instead.
fun.path	Deprecated. Use path.format.

Details

The function handles both single plot objects and lists of plots with flexible output options:

Single plots:

- Written to file with specified format(s)
- Can generate multiple formats simultaneously
- Can generate multiple canvas sizes simultaneously

Lists of plots:

- If onefile = TRUE: creates a single multi-page PDF
- If onefile = FALSE: creates separate files (numbered or named)
- Automatic naming based on list element names or indices

Value

Returns NULL invisibly. Files are written as a side effect. If show = TRUE, plots are also displayed.

Canvas Sizes

Predefined canvas sizes (see canvasSize):

- "standard": 7.2 × 5.4 inches
- "wide": 9.6 × 5.4 inches
- "wide-screen": 18.6 × 9 inches
- "square": 5.4 × 5.4 inches
- "A4": 5.4 × 7.2 inches
- "letter": 6.5 × 9 inches

Custom sizes can be specified as list(width = w, height = h) where w and h are in inches.

Stamping

When `script` is provided, a caption is added to the plot using `ggstamp`. The stamp typically includes:

- Script file name
- Timestamp (unless `time = ""`)
- Model identifier (if `model` is provided)

The stamp is added as a ggplot2 caption and will appear at the bottom of the plot.

Trace Attributes

If `plot` has traceit attributes (added via `traceit`), those attributes will be automatically extracted and used, unless explicitly overridden by arguments passed to `ggwrite`.

See Also

`ggstamp` for adding stamps to plots, `canvasSize` for canvas size options, `writer` for a unified interface to write various object types, `pathFormat` for organizing output directories, `traceit` for adding traceit attributes, `png`, `pdf` for graphics devices

Other Plotting: `canvasSize()`, `ggstamp()`, `ggwater()`

Examples

```
library(ggplot2)
writeOutput <- FALSE
data(ChickWeight)
p1 <- ggplot(ChickWeight, aes(Time, weight, group = Chick, colour = factor(Diet))) +
  geom_line()

# View plot on screen
ggwrite(p1)

# Save plot with stamping
script <- "analysis.R"
ggwrite(p1, script = script, canvas = "wide",
        file = "myplot1.png", save = writeOutput)

# Save in multiple formats
ggwrite(p1, file = "myplot1.png",
        formats = c("png", "pdf"),
        save = writeOutput)

# Multiple canvas sizes
ggwrite(p1, file = "myplot1.png",
        canvas = list(small = "standard", large = "wide"),
        save = writeOutput)
# Creates: myplot1_small.png, myplot1_large.png

# List of plots to separate files
p2 <- ggplot(ChickWeight, aes(Time, weight)) + geom_point()
```

```
plots <- list(lines = p1, points = p2)
ggwrite(plots, file = "chickweight.png",
        use.names = TRUE, save = writeOutput)
# Creates: chickweight_lines.png, chickweight_points.png

# List of plots to single PDF
ggwrite(plots, file = "chickweight.pdf",
        onefile = TRUE, save = writeOutput)
# Creates: chickweight.pdf (2 pages)

# Organize output using a %script-based directory structure
ggwrite(p1, file = "chickweight_lines.png",
        script = script, path.format = "%script/%name",
        save = writeOutput)
# Creates: analysis/chickweight_lines.png
```

`is.gg` *Check if an object is a ggplot or gtable*

Description

This function tests whether an object is a ggplot2 plot object or a gtable (grid table) object.

Usage

```
is.gg(x)
```

Arguments

`x`	An object to test

Value

Logical. TRUE if x inherits from "ggplot" or "gtable", FALSE otherwise.

Examples

```
## Not run:
library(ggplot2)
p <- ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point()
is.gg(p)  # TRUE
is.gg(mtcars)  # FALSE

## End(Not run)
```

is.listnotplot *Check if an object is a list but not a plot, data frame, or flextable*

Description

This function tests whether an object is a list that is NOT a ggplot/gtable, data frame, or flextable object. This is useful for identifying nested list structures that contain multiple outputs.

Usage

```
is.listnotplot(x)
```

Arguments

x	An object to test

Value

Logical. TRUE if x is a list but not a ggplot, gtable, data.frame, or flextable object; FALSE otherwise.

Examples

```
## Not run:
library(ggplot2)
p <- ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point()
is.listnotplot(p)  # FALSE (it's a ggplot)
is.listnotplot(mtcars)  # FALSE (it's a data.frame)
is.listnotplot(list(a = 1, b = 2))  # TRUE (it's a plain list)

## End(Not run)
```

lwrite *Write lists of outputs with flexible organization and naming*

Description

Writes nested lists containing plots, tables, and data frames to files with intelligent handling of list structure, automatic naming, and optional subdirectory organization. This is particularly useful for organizing multiple related outputs from a single analysis or model run.

Usage

```
lwrite(
  list,
  model,
  dir = ".",
  path.format,
  lists.as.subdirs = FALSE,
  name.all.pdf = "main_all.pdf",
  structure,
  ...
)
```

Arguments

list	A list (potentially nested) containing output objects: ggplot plots, flextable tables, and/or data.frames. The list can be arbitrarily nested, and names at each level will be used to construct output file names.
model	Character string or model object specifying the model name/identifier. Used for organizing output directories and stamping. If a list (model object) is provided, the $label element is extracted.
dir	Character string specifying the base output directory. Default is "." (current directory).
path.format	Character string or function defining the output path format. Passed to pathFormat to create a path-generating function. Examples: "%model/%name_%model" (default), "%name_%model", "%script/%name_%model". See pathFormat for details.
lists.as.subdirs	Logical. If TRUE, nested lists create subdirectories rather than being incorporated into file names. Default is FALSE. • FALSE: nested list names become part of file names (e.g., list1_item1.png) • TRUE: nested lists create subdirectories (e.g., list1/item1.png)
name.all.pdf	Character string specifying the filename to use when writing all plots to a single PDF file. Only used when the file name is empty and plots are being combined into one PDF. Default is "main_all.pdf".
structure	Deprecated. Use path.format.
...	Additional arguments passed to writer and subsequently to the specific writer functions (ggwrite, ftwrite, or datwrite). Common arguments include: • script: Path to script file for stamping • time: Timestamp for stamping • formats.gg: File formats for plots (default "png") • formats.ft: File formats for tables (default "png") • formats.data: File formats for data frames (default "rds") • canvas: Canvas size for plots • quiet: Suppress messages • stamp.format, stamp.funs: Stamp text format, see createStamp

- `stamp.gg.style`, `stamp.ft.style`: Stamp styling for plots and tables, see `writer`

Details

The function processes lists recursively with the following behavior:

1. Separates plot objects from non-plot objects
2. Combines plots into a single multi-page or multi-file output (see `ggwrite`)
3. Writes each non-plot object individually
4. Handles nested lists according to `lists.as.subdirs`

Value

Invisibly returns a list of results from the `writer` calls. The structure mirrors the input list structure.

List Structure Handling

The function intelligently flattens nested lists while preserving meaningful names:

- Named list elements use their names in output file names
- Plots in the same list are combined (behavior depends on `ggwrite` settings)
- Non-plot objects (tables, data frames) are written individually
- Nesting deeper than 2 levels will cause an error

File Naming

Output file names are constructed from:

- The name at each level of the list hierarchy
- The `model` identifier (if provided and if `path.format` uses it)
- Automatic numbering for unnamed elements

For example, with `lists.as.subdirs` = `FALSE`:

```
list(
  section1 = list(
    plot1 = p1,
    table1 = ft1
  )
)
```

produces: `section1_plot1.png`, `section1_table1.png`

With `lists.as.subdirs` = `TRUE`, the same structure produces: `section1/plot1.png`, `section1/table1.png`

Plot Handling

All `ggplot` objects within the same list level are collected and written together using `ggwrite`. This allows creating multi-page PDFs or sequentially numbered PNG files from a list of plots.

See Also

writer for the underlying writing function, ggwrite for plot-specific options, ftwrite for table-specific options, datwrite for data frame writing, pathFormat for path format options, is.listnotplot for list detection logic

Examples

```
## Not run:
library(ggplot2)
library(flextable)

# Create some outputs
p1 <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
p2 <- ggplot(mtcars, aes(hp, mpg)) + geom_point()
ft1 <- flextable(head(mtcars))
df1 <- mtcars

# Simple list
outputs <- list(
  scatter_wt = p1,
  scatter_hp = p2,
  summary_table = ft1,
  data = df1
)

lwrite(outputs, model = "run001", dir = "outputs",script="myscript")
# Creates in outputs/run001/:
#   scatter_wt_run001.png
#   scatter_hp_run001.png
#   summary_table_run001.png
#   data_run001.rds

# Nested list with subdirectories
nested_outputs <- list(
  diagnostics = list(
    gof1 = p1,
    gof2 = p2
  ),
  tables = list(
    summary = ft1
  )
)

lwrite(nested_outputs,
       model = "run001",
       dir = "outputs",
       lists.as.subdirs = TRUE)
# Creates:
#   outputs/run001/diagnostics/gof1_run001.png
#   outputs/run001/diagnostics/gof2_run001.png
#   outputs/run001/tables/summary_run001.png
```

```
# With stamping
lwrite(outputs,
       model = "run001",
       dir = "outputs",
       script = "analysis.R",
       time = Sys.time())

## End(Not run)
```

---

padZeros *Pad Numbers with Leading Zeros*

Description

Converts numeric values to character strings with leading zeros to achieve a specified total width.

Usage

```
padZeros(x, nchars)
```

Arguments

x	A numeric vector of integers to be padded with leading zeros.
nchars	An integer specifying the total number of characters in the output string (including leading zeros).

Value

A character vector of the same length as x, with each element padded with leading zeros to reach nchars characters.

Examples

```
# Pad single digit numbers to 3 characters
padZeros(5, 3)
# Returns: "005"

# Pad a vector of numbers
padZeros(c(1, 10, 100), 4)
# Returns: c("0001", "0010", "0100")

# Useful for creating sequential file names
paste0("file_", padZeros(1:5, 3), ".txt")
# Returns: c("file_001.txt", "file_002.txt", ..., "file_005.txt")

# Or append to an existing file name
NMdata::fnAppend("file.txt", padZeros(1:5, 3), collapse=NULL)
```

pathFormat *Create a function to derive output file paths from a path format*

Description

Generates a function that constructs file paths according to a path format (a template with placeholders). This is particularly useful in modeling workflows where outputs need to be organized by model name, with consistent naming conventions across files.

Usage

```
pathFormat(
  path.format = "%name",
  dir,
  model = NULL,
  script = NULL,
  path.funs = NULL,
  structure
)
```

Arguments

path.format	A character string specifying a path format template, or a custom function. If a function is provided, it is returned without modification. Templates are specified using %model, %script, and %name placeholders, separated by / to indicate directory levels. The last /-separated component is treated as the file name pattern; any earlier components are directory names. %model is replaced by the model name (with any file extension stripped), %script by the script name (with any file extension stripped), and %name by name (also with its extension stripped, re-attached at the end). Any other characters (e.g. -, _) are kept as literal separators. Examples: "%model/%name_%model" Creates paths like model_name/file_model_name.ext. Example: "run001/gof_run001.png". "%name_%model" Creates paths like file_model_name.ext with no model subdirectory. Example: "gof_run001.png" "%model/%model-%name" Creates paths like model_name/model_name-file.ext. Example: "run001/run001-gof.png" "%model-%name" Creates paths like model_name-file.ext with no model subdirectory. Example: "run001-gof.png" "%script/%name_%model" Creates paths like script_name/file_model_name.ext. Example: "myscript/gof_run001.png" Any other combination of %model, %script, %name, literal text, and / is supported - e.g. "archive/%model/plots/%name". Default is "%name", i.e. name is used as is (relative to dir).
dir	Character string specifying the base directory for all outputs. This is prepended to all generated paths. Default is NULL (current directory).

`model` Default value for the `model` argument of the generated function. Used whenever the generated function is called without explicitly supplying `model`. Default is `NULL`.

`script` Default value for the `script` argument of the generated function. Used whenever the generated function is called without explicitly supplying `script`. Default is `NULL`.

`path.funs` Optional named list of functions used to preprocess `model` and/or `script` before they are substituted into `path.format`. Supported names are `model` and `script`. Each function takes the value as its single argument and returns a character string. Elements not supplied keep the default, which is `function(x) fnExtension(basename(x), "")` (strip directory and file extension). Note that a supplied function replaces the default entirely, so e.g. extension stripping must be included if still wanted. Example: `path.funs = list(script = function(x) substr(basename(x), 1, 4))`. Default is `NULL`.

`structure` Deprecated. Use `path.format`.

Details

The returned function accepts the following arguments:

`name` The base file name (typically without extension)

`model` A model identifier (character string or model object). If a model object (list) is provided, the `$mod` element is extracted and used as the model name. The file extension is automatically removed from the model name (e.g., "model.lst" becomes "model").

`script` A script identifier (character string, typically a script file name). The file extension is automatically removed (e.g., "myscript.R" becomes "myscript").

`subdir` Optional subdirectory. Inserted between the directory components of `path.format` and the file name component.

`dir` Base directory. Overrides the `dir` argument passed to `pathFormat`.

`quiet` If `TRUE`, suppress messages about ignored `model`/`script` arguments. Default `FALSE`.

The generated function handles missing `model`/`script` arguments gracefully - if `path.format` does not contain `%model` (or `%script`), the corresponding argument is optional and ignored (with a message if supplied anyway). If `path.format` does contain `%model` (or `%script`), the corresponding argument is required.

Value

A function with signature `function(name, model, script, subdir, dir, quiet = FALSE)` that generates file paths according to the specified path format. If `path.format` is already a function, it is returned unchanged.

See Also

`writer` for using path formats in writing functions, `ggwrite`, `ftwrite`, `lwrite` for functions that accept `fun.path` arguments

Examples

```
# Create a path function (default format)
path_fun <- pathFormat(path.format = "%model/%name_%model")

# Use it to generate paths
path_fun(name = "gof1.png", model = "run001")
# Returns: "run001/gof1_run001.png"

path_fun(name = "gof1.png", model = "run001", subdir = "diagnostics")
# Returns: "run001/diagnostics/gof1_run001.png"

# With base directory
path_fun2 <- pathFormat(path.format = "%model/%model-%name", dir = "outputs")
path_fun2(name = "gof1.png", model = "run001")
# Returns: "outputs/run001/run001-gof1.png"

# Different format
path_fun3 <- pathFormat(path.format = "%model-%name")
path_fun3(name = "gof1.png", model = "run001")
# Returns: "run001-gof1.png"

# Fully custom format
path_fun3b <- pathFormat(path.format = "archive/%model/plots/%name")
path_fun3b(name = "gof1.png", model = "run001")
# Returns: "archive/run001/plots/gof1.png"

# Format based on script name
path_fun3c <- pathFormat(path.format = "%script/%name_%model")
path_fun3c(name = "gof1.png", model = "run001", script = "myscript")
# Returns: "myscript/gof1_run001.png"

# Supplying default model/script to pathFormat() itself
path_fun3d <- pathFormat(path.format = "%script/%name_%model",
                         model = "run001", script = "myscript")
path_fun3d(name = "gof1.png")
# Returns: "myscript/gof1_run001.png"

# Explicit arguments to the generated function override the defaults
path_fun3d(name = "gof1.png", model = "run002")
# Returns: "myscript/gof1_run002.png"

# Custom preprocessing of script (and/or model) via path.funs
path_fun3e <- pathFormat(path.format = "%script/%name_%model",
                        path.funs = list(script = function(x) substr(basename(x), 1, 4)))
path_fun3e(name = "gof1.png", model = "run001", script = "myscript.R")
# Returns: "mysc/gof1_run001.png"

# Custom function (passed through unchanged)
my_fun <- function(name, model, subdir) {
  paste0("custom/", model, "/", name)
}
path_fun4 <- pathFormat(path.format = my_fun)
```

```
identical(path_fun4, my_fun)  # TRUE
```

`stampFlextab` *Stamp and write flextab objects (deprecated - use ftstamp)*

Description

Deprecated - use `ftstamp()` instead.

Usage

```
stampFlextab(ft, file, script, bg = "#ffffff", time)
```

Arguments

`ft`	a flextable object
`file`	The file that the flextable will be written to (no file is written by this function)
`script`	path to script - will be pasted as caption.
`bg`	Default bacground colour is #ffffff.
`time`	The default is to insert a time stamp taken from result of Sys.time(). Using the time argument you can overrule this by setting a fixed string instead. Use "" to omit.

Value

A flextable object with the stamp added as a footer line.

`traceit` *Add tracing attributes to R objects for automated output writing*

Description

The `traceit` family of functions attaches metadata to R objects that can be used later by writer functions to automatically determine output settings. This allows you to "trace" or "tag" objects with information about how they should be written, then write them later without having to re-specify those settings.

Usage

```
traceit(x, ...)

## S3 method for class 'ggplot'
traceit(x, ..., quite = TRUE)

## S3 method for class 'flextable'
traceit(x, ..., quite = TRUE)

## S3 method for class 'data.frame'
traceit(x, ..., quite = TRUE)

## S3 method for class 'traceit'
traceit(x, ..., quite = TRUE)
```

Arguments

x	An R object to attach trace attributes to. Can be a ggplot, flextable, or data.frame object.
...	Named arguments that will be stored as trace attributes. For ggplot objects, valid arguments match those in ggwrite. For flextable objects, valid arguments match those in ftwrite. For data.frame objects, valid arguments match those in NMwriteData.
quite	Logical. If TRUE, suppresses messages. Default is TRUE. Note: this parameter name appears to be a typo and should likely be "quiet".

Details

The traceit function is a generic that dispatches to specific methods based on the class of x:

- traceit.ggplot: For ggplot objects
- traceit.flextable: For flextable objects
- traceit.data.frame: For data frames
- traceit.traceit: For objects that already have traceit attributes (allows updating existing trace information)

When an object is traced, two attributes are added:

- args: A list of arguments to be used when writing the object
- class: The class vector is prepended with "traceit" to indicate the object has been traced

Arguments passed to traceit are filtered to include only those that are valid for the corresponding writer function. Invalid arguments are silently ignored.

When calling traceit on an already-traced object, the new arguments are merged with existing trace arguments using modifyList, with new arguments taking precedence.

Value

Returns x invisibly with trace attributes attached. The object can be used normally and the trace attributes will be preserved.

Extracting Trace Attributes

To extract trace attributes from an object, use argsFromTrace.

See Also

ggwrite for writing ggplot objects, ftwrite for writing flextable objects, NMwriteData for writing data frames, argsFromTrace for extracting trace attributes, writer for a unified writing interface

Examples

```
## Not run:
library(ggplot2)
library(flextable)

# Create a plot and trace it with output settings
p1 <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
p1_traced <- traceit(p1, canvas = "wide", formats = c("png", "pdf"))

# The traced plot can be written later without re-specifying settings
# writer() will automatically use the traced arguments

# Trace a flextable
ft1 <- flextable(head(mtcars))
ft1_traced <- traceit(ft1, formats = "docx")

# Update trace attributes on an already-traced object
p1_traced <- traceit(p1_traced, canvas = "standard")  # overrides previous canvas

# Trace a data frame for NMwriteData
df1 <- data.frame(ID = 1:10, DV = rnorm(10))
df1_traced <- traceit(df1, file = "mydata.csv")

## End(Not run)
```

---

writer *Write various R objects to files with flexible formatting*

Description

A unified interface for ggwrite, ftwrite, and NMwriteData that automatically detects object type and applies the appropriate writing function with consistent file naming and stamping conventions.

Usage

```
writer(
  x,
  file,
  formats.ft,
  formats.gg,
  formats.data,
  path.format = NULL,
  subdir = NULL,
  stamp.format = NULL,
  stamp.funs = NULL,
  stamp.gg.style = NULL,
  stamp.ft.style = NULL,
  fun.path = NULL,
  ...
)
```

Arguments

x	Object to write. Can be a ggplot, flextable, data.frame, or list of such objects. Lists can be arbitrarily nested and will be processed according to object type.
file	Character string specifying the base file name (without extension). The actual extension will be determined by the formats.* arguments.
formats.ft	Character vector of file formats for flextable objects. Default is "png". Supported formats include: "png", "docx", "pptx", "html". See ftwrite for details.
formats.gg	Character vector of file formats for ggplot objects. Default is "png". Supported formats: "png", "pdf". See ggwrite for details.
formats.data	Character vector of file formats for data frame objects. Default is "rds". This is passed to NMwriteData as the format.write argument. Common formats include: "rds", "csv", "fst".
path.format	Character string or function defining the output path. A template using %model, %script, and %name placeholders, or a function as returned by pathFormat. Passed to pathFormat. Default (NULL) uses file as is.
subdir	Character string specifying a subdirectory within the path structure. This is passed to the path function generated from path.format.
stamp.format	A template for the stamp text using the placeholders %script, %time, %model, and %file, with "\n" for line breaks. Default is "%file\n%script\n%model\n%time". See createStamp.
stamp.funs	Optional named list of functions (elements script, time, model, file) used to preprocess the values before they are inserted into stamp.format. See createStamp.
stamp.gg.style	A list of formatting options for the stamp text, passed to ggstamp. For ggplot objects, elements accepted by ggplot2::element_text() (e.g. size, colour, hjust).
stamp.ft.style	A list of formatting options for the stamp, passed to ftstamp. Valid elements: size, align, space, bg, color.

fun.path	Deprecated. Use path.format.
...	Additional arguments passed to the specific writer functions (ggwrite, ftwrite, or NMwriteData). Common arguments include: • script: Path to the script file (for stamping) • time: Time stamp (for stamping) • model: Model identifier (for stamping and path construction) • save: Logical, whether to save the output (default TRUE) • show: Logical, whether to show/print the output (default !save) • canvas: Canvas size for plots (see canvasSize) • quiet: Logical, suppress messages (default FALSE)

Details

The function dispatches to appropriate writer functions based on object class:

- flextable objects → ftwrite
- data.frame objects → datwrite (which wraps NMwriteData)
- ggplot/gtable objects or lists → ggwrite

For lists of objects, each element is written separately with appropriate naming. The function intelligently handles nested lists by flattening them and constructing meaningful file names from the list structure.

File paths are constructed using path.format, which can organize outputs by model, file name, or custom patterns. This is particularly useful in modeling workflows where outputs need to be organized by model number or name.

Value

Invisibly returns the result from the called writer function. The specific return value depends on which writer function was dispatched to.

Object Type Detection

The function determines object type in the following order:

1. Check if flextable
2. Check if data.frame
3. Check if ggplot or gtable (using is.gg)
4. Check if list (and not a plot object)
5. If none of the above, throw an error

Trace Attributes

If the object x has traceit attributes (added via traceit), those attributes will be automatically extracted and used by the writer function, unless explicitly overridden by arguments passed to writer.

See Also

ggwrite for writing ggplot objects, ftwrite for writing flextable objects, datwrite for writing data frames, lwrite for writing lists with subdirectory organization, pathFormat for path format options, traceit for adding traceit attributes to objects

Examples

```
## Not run:
library(ggplot2)
library(flextable)

# Write a ggplot
p1 <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
writer(p1, file = "myplot", formats.gg = c("png", "pdf"))

# Write a flextable
ft1 <- flextable(head(mtcars))
writer(ft1, file = "mytable", formats.ft = c("png", "docx"))

# Write a data frame
writer(mtcars, file = "mydata", formats.data = "csv")

# Use with model organization
writer(p1, file = "gof", model = "run001", path.format = "%model/%name_%model")
# Writes to: run001/gof_run001.png

# Write with stamping
writer(p1, file = "myplot",
       script = "analysis.R",
       time = Sys.time(),
       model = "run001")

## End(Not run)
```

Index

∗ **Plotting**
    canvasSize, 2
    ggstamp, 8
    ggwater, 10
    ggwrite, 11

agg_png, *12*
argsFromTrace, *26*
arrangeGrob, *11*

canvasSize, 2, *9, 10*, *12–14*, *28*
createStamp, *4*, *6*, *9*, *13*, *17*, *27*

datwrite, 3, *17*, *19*, *28, 29*

ftstamp, 4, *6, 7*, *27*
ftwrite, 5, *17*, *19*, *22*, *25–29*
ftwriteOne, *7*

ggstamp, *3*, 8, *10*, *13, 14*, *27*
ggwater, *3*, *9*, 10, *14*
ggwrite, *3*, *9, 10*, 11, *17–19*, *22*, *25–29*
grid.arrange, *11*

is.gg, 15, *28*
is.listnotplot, 16, *19*

lwrite, 16, *22*, *29*

modifyList, *25*

NMwriteData, *25–28*

padZeros, 20
pathFormat, *3*, *6, 7*, *12*, *14*, *17*, *19*, 21, *27*, *29*
pdf, *12*, *14*
png, *12*, *14*

save_as_docx, *7*
save_as_html, *7*
save_as_image, *7*
save_as_pptx, *7*
stampFlextab, 24

traceit, *14*, 24, *28, 29*

writeFlextab *(*ftwrite*)*, 5
writer, *7*, *14*, *17–19*, *22*, 26, *26*